Engineering

Andreas Mueller (NYU Center for Data Science, scikit-learn)

Goals

Achievements

Methods

Challenges

Goals

Goal:
High quality, easy to use machine learning library.

Goal:
High quality, easy to use machine learning library.
Keep it usable, keep it maintainable.

Simple things should be simple, complex things should be possible.

Alan Kay

Non-Goals

Non-programmatic interfaces

Algorithm development

Cutting edge algorithms

Structured, online, or reinforcement learning.

“I thought it was more like CRAN”

Achievements

Unwatch 532 | Unstar 4,429 | Fork 2,546

Downloads (All Versions):

4916 downloads in the last day

32371 downloads in the last week

106380 downloads in the last month

Cited by 942

We've been using it quite a lot for music recommendations at Spotify and I think it's the most well-designed ML package I've seen so far.

- spotify

scikit-learn in one word: Awesome.

- machinalis

I'm constantly recommending that more developers and scientists try scikit-learn.

- lovely

The documentation is really thorough, as well, which makes the library quite easy to use.

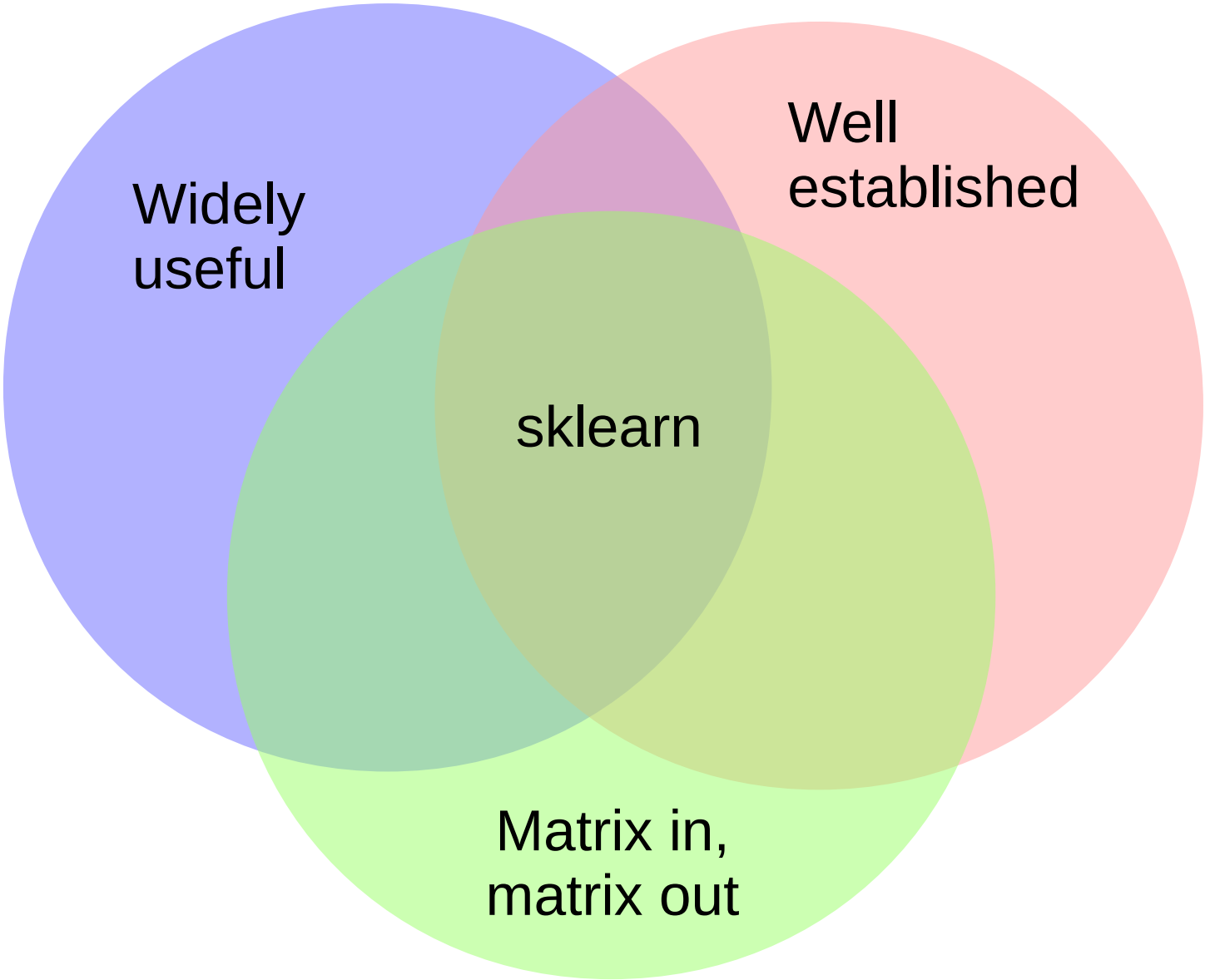
- OkCupid

scikit-learn makes doing advanced analysis in Python accessible to anyone.

- yhat

Methods

Scoping


Widely useful
Well established
sklearn
Matrix in, matrix out

Simplicity

```
est = Est()
est.fit(X_train, y_train)
est.score(X_test, y_test)
```

Consistency

```
grid = GridSearchCV(svm,param_grid)
grid.fit(X_train, y_train)
grid.score(X_test, y_test)
```

Sensible Defaults

Everything is default constructible!

```
for clf in [KneighborsClassifier(),
            SVC(),
            DecisionTreeClassifier(),
            RandomForestClassifier(),
            AdaBoostClassifier(),
            GaussianNB(),
            LDA(),
            QDA()]:
  clf.fit(X_train, y_train)
  print(clf.score(X_test, y_test))
```

Common Tests

```
classifiers = all_estimators(type_filter='classifier')
for name, Classifier in classifiers:
    # test classfiers can handle non-array data
    yield check_classifier_data_not_an_array, name, Classifier
    # test classifiers trained on a single label
    # always return this label
    yield check_classifiers_one_label, name, Classifier
    yield check_classifiers_classes, name, Classifier
    yield check_classifiers_pickle, name, Classifier
    yield check_estimators_partial_fit_n_features, name, Classifier
```

Flat Class Hierarchy, Few Types

- Numpy arrays / sparse matrices
- Estimators
- [Cross-validation objects]
- [Scorers]

No Framework

“This looks frameworkish.” means “try again.”

Avoid Code

- Code rots!
- Hail all code deleters!

Three-Way Documentation

1.9. Ensemble methods

The goal of **ensemble methods** is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

Two families of ensemble methods are usually distinguished:

- In **averaging methods**, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.

  Examples: *Bagging methods*, *Forests of randomized trees*, ...
- By contrast, in **boosting methods**, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble.

  Examples: *AdaBoost*, *Gradient Tree Boosting*, ...

`sklearn.ensemble`.RandomForestClassifier

class `sklearn.ensemble.` **`RandomForestClassifier`** (*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

Parameters:

n_estimators : integer, optional (default=10)

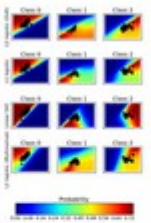
The number of trees in the forest.

criterion : string, optional (default="gini")

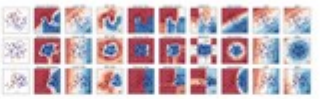
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

Examples

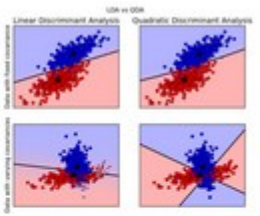
Recognizing hand-written digits



Plot classification probability



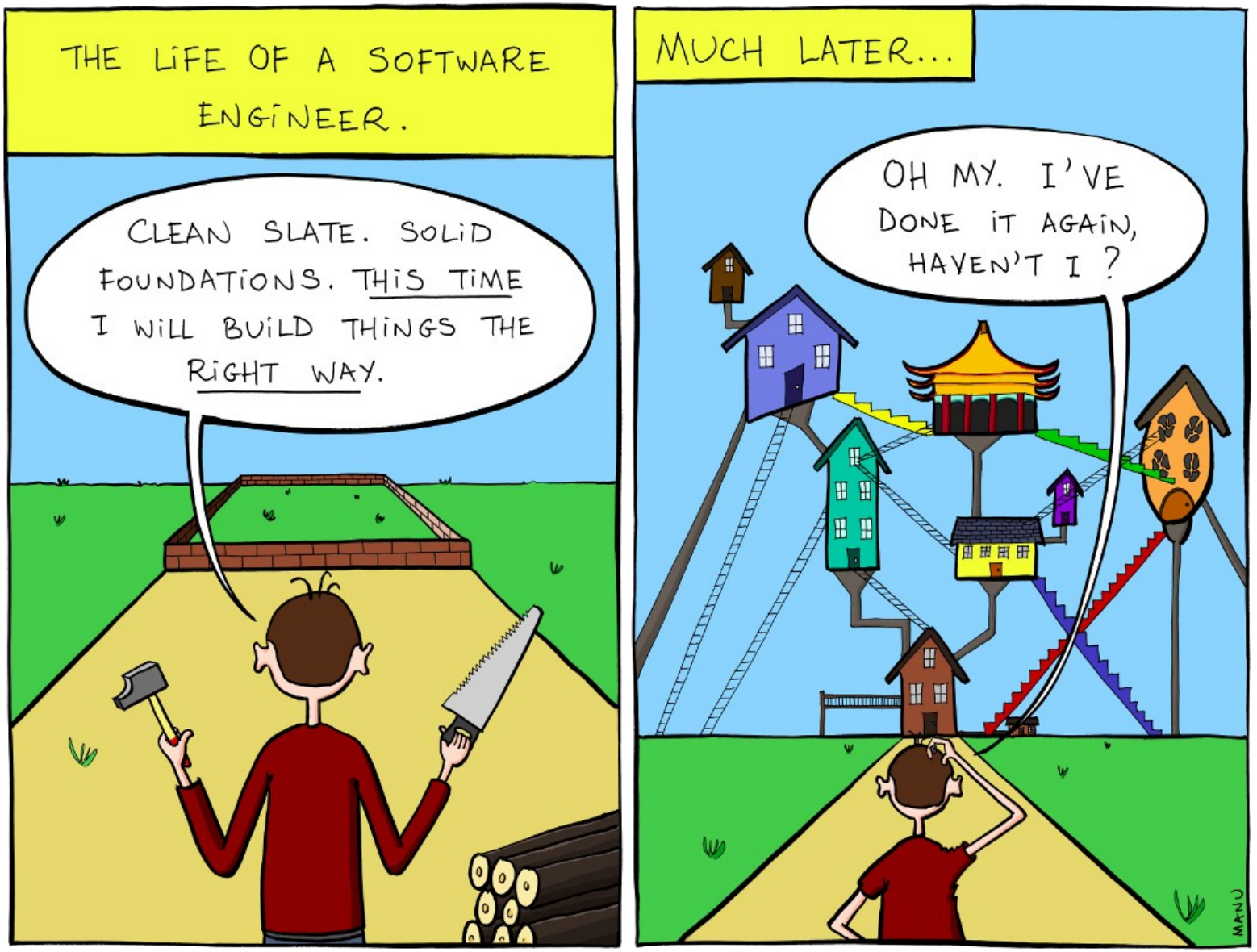
Classifier comparison



Linear and Quadratic Discriminant Analysis with confidence ellipsoid

Challenges

Feature Creep



Multi-Platform Support

- Linux / Mac / Windows / Solaris (no kidding)
- 32bit / 64bit
- Python2.6 / Python2.7 / Python 3.4
- GCC, Clang, MSVC
- Blas dependency...

- And we want “one click” install

Two Language Problem

Two Language Problem

C / C++

Backward compatibility

```
from sklearn.cross_validation import Bootstrap
Bootstrap(10)

sklearn/cross_validation.py:685:
DeprecationWarning: Bootstrap will no longer
be supported as a cross-validation method as of
version 0.15 and will be removed in 0.17.
```

Backward compatibility

```
andy@csd-amueller  ~/checkout/scikit-learn  ‹master›
$ git grep DeprecationWarning | grep -v test | wc -l
39
```

Backward compatibility

```
>>> import pickle
>>> s = pickle.dumps(clf)
>>> clf2 = pickle.loads(s)
>>> clf2.predict(X[0])
array([0])
>>> y[0]
0
```

Correctness Testing

?

Project Size

193 Open ✔ 2,083 Closed

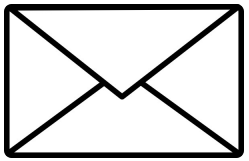
337 Open ✔ 1,318 Closed

In a Nutshell, scikit learn...

··· has had 17,356 commits made by 424 contributors
representing 433,767 lines of code

@t3kcit

@amueller



amueller@nyu.edu